COMPUTER SCIENCE

A.K. Watson Hall, 203.432.1246
http://cpsc.yale.edu
M.S., M.Phil., Ph.D.

Chair
Zhong Shao

Director of Graduate Studies
Vladimir Rokhlin (108 AKW, 203.432.1278, vladimir.rokhlin@yale.edu)


Senior Lecturers James Glenn, Stephen Slade

Lecturers Timothy Barron, Andrew Bridy, † Ozan Erat, Jay Lim, Cody Murphey, Scott Petersen, Brad Rosen, Andrew Sherman, * Inyoung Shin, Alan Weide, Cecillia Xie

* A secondary appointment with primary affiliation in another department or school.
† A joint appointment with another department.

FIELDS OF STUDY
Algorithms and computational complexity, artificial intelligence, data networking, databases, graphics, machine learning, programming languages, robotics, scientific computing, security and privacy, and systems.

RESEARCH FACILITIES
The department operates a high-bandwidth, local-area computer network-based mainly on distributed workstations and servers, with connections to worldwide networks. Workstations include Dell dual-processor PCs (running Linux or Windows/XP). Laboratory contains specialized equipment for graphics, vision, and robotics research. Various printers, including color printers, as well as image scanners, are also available. The primary educational facility consists of thirty-seven PC workstations supported by a large Intel PC server. This facility is used for courses and unsponsored research by Computer Science majors and first-year graduate students. Access to computing,
through both the workstations and remote login facilities, is available to everyone in the department.

**SPECIAL REQUIREMENTS FOR THE PH.D. DEGREE**

There is no foreign language requirement. To be admitted to candidacy, a student must (1) pass ten courses (including CPSC 690 and CPSC 691) with at least two grades of Honors, the remainder at least High Pass, including three advanced courses in an area of specialization; (2) take six advanced courses in areas of general computer science; (3) successfully complete a research project in CPSC 690, CPSC 691, and submit a written report on it to the faculty; (4) pass a qualifying examination in an area of specialization; (5) be accepted as a thesis student by a regular department faculty member; (6) serve as a teaching assistant for two terms; and (7) submit a written dissertation prospectus, with a tentative title for the dissertation. Grades of Pass will not count toward the Ph.D. To satisfy the distribution requirement (requirement 2 above), the student must take one course in programming languages or systems, one programming-intensive course, two theory courses, and two in application areas. In order to gain teaching experience, all graduate students are required to serve as teaching assistants for two terms during their first three years of study. All requirements for admission to candidacy must be completed prior to the end of the third year. In addition to all other requirements, students must successfully complete CPSC 991, Ethical Conduct of Research, prior to the end of their first year of study. This requirement must be met prior to registering for the second year of study.

**MASTER’S DEGREES**

**M.Phil.** See Degree Requirements under Policies and Regulations.

**M.S. (en route to the Ph.D.)** To qualify for the M.S., the student must pass eight courses at the 500 level or above from an approved list. An average grade of at least High Pass is required, with at least one grade of Honors.

**Terminal Master’s Degree Program** Students may also be admitted to a terminal master’s degree program directly. There are two options for the terminal master’s degree:

- **Terminal Master’s Degree Program (coursework-only option)** The requirements are the same as for the M.S. en route to the Ph.D. This program is normally completed in one year, but a part-time program may be spread over as many as four years.

- **Terminal Master’s Degree Program (thesis option)** To qualify for the M.S. thesis option the student must (1) pass six courses at the 500 level or above from an approved list with an average grade of at least High Pass and with at least one grade of Honors; (2) complete a research thesis, generally in the second year; and (3) serve as a teaching assistant for four terms. This program is normally completed in two years.

Please use the links provided for additional information about the department, faculty, courses, and facilities online; You may also reach out to the departmental registrar at cs-admissions@cs.yale.edu.
COURSES

CPSC 513a, Computer System Security  Timothy Barron
Overview of the principles and practice behind analyzing, designing, and implementing secure computer systems. The course covers problems that have continued to plague computer systems for years as well as recent events and research in this rapidly evolving field. Students learn to think from the perspective of an adversary, to understand systems well enough to see how their flaws could be exploited, and to consequently defend against such exploitation. The course offers opportunities for hands-on exploration of attacks and defenses in the contexts of web applications, networks, and system-level software. It also addresses ethical considerations and responsibilities associated with security research and practice.

CPSC 519a, Full Stack Web Programming  Jay Lim
This course introduces students to a variety of advanced software engineering and programming techniques in the context of full-stack web programming. The focus of the course includes both client- and server-side programming (and database programming), client/server communication, user interface programming, and parallel programming.

CPSC 520b / ENAS 820b, Computer Architecture  Abhishek Bhattacharjee
This course offers a treatment of computer architectures for high-performance and power/energy-efficient computer systems. Topics include the foundations of general-purpose computing, including instruction set architectures, pipelines, superscalar and out-of-order execution, speculation, support for precise exceptions, and simultaneous multi-threading. We also cover domain-specific hardware (e.g., graphics processing units), and ongoing industry efforts to elevate them to the status of first-class computing units. In tandem, we cover topics relevant to both general-purpose and domain-specific computing, including memory hierarchies, address translation and virtual memory, on-chip networks, machine learning techniques for resource management, and coherence techniques. If time permits, we study the basics of emerging non-classical computing paradigms like neuromorphic computing. Overall, this course offers insights on how the computing industry is combating the waning of traditional technology scaling via acceleration and heterogeneity. Prerequisites: Courses similar to CPSC 323, 223, and 202. This is a programming-intensive course, so comfort with large programming projects is essential.

CPSC 521b, Compilers and Interpreters  Jay Lim
Compiler organization and implementation: lexical analysis, formal syntax specification, parsing techniques, execution environment, storage management, code generation and optimization, procedure linkage, and address binding. The effect of language-design decisions on compiler construction.

CPSC 522a, Operating Systems  Zhong Shao
The design and implementation of operating systems. Topics include synchronization, deadlocks, process management, storage management, file systems, security, protection, and networking.

CPSC 524a, Parallel Programming Techniques  Andrew Sherman
Practical introduction to parallel programming, emphasizing techniques and algorithms suitable for scientific and engineering computations. Aspects of processor and machine architecture. Techniques such as multithreading, message passing, and data parallel
computing using graphics processing units. Performance measurement, tuning, and debugging of parallel programs. Parallel file systems and I/O.

**CPSC 526b, Building Distributed Systems**  Y. Richard Yang
Ubiquitous services such as Google, Facebook, and Amazon run on the back of massive distributed systems. This course covers the fundamental principles, abstractions, and mechanisms that inform the design of such systems, as well as the practical details of real-world implementations. Technical topics covered include properties such as consistency, availability, durability, isolation, and failure atomicity; as well as protocols such as RPC, consensus, consistent hashing, and distributed transactions. The final project involves implementing a real-world distributed service.

**CPSC 529a, Principles of Computer System Design**  Lin Zhong
Humans are stupid; computers are limited. Yet a collaboration of humans and computers has led to ever more powerful and complex computer systems. This course examines the limitations of humans and computers in this endeavor and how they shape the design, implementation, and evaluation of computer systems. It surveys the empirical knowledge reported by scholars and practitioners who overcome such limitations. The lectures, reading assignments, and classroom discussions travel through psychology and philosophy and revisit important results from theoretical computer science, with a goal of elucidating the rationales behind the best practices in computer systems research and development. Prerequisite: CPSC 323 or equivalent. Students should have the ability to write significant system programs in at least one system programming language (e.g., C, C++ and Rust).

**CPSC 531a, Computer Music: Algorithmic and Heuristic Composition**  Scott Petersen
Study of the theoretical and practical fundamentals of computer-generated music. Music and sound representations, acoustics and sound synthesis, scales and tuning systems, algorithmic and heuristic composition, and programming languages for computer music. Theoretical concepts are supplemented with pragmatic issues expressed in a high-level programming language.

**CPSC 532b, Computer Music: Sound Representation and Synthesis**  Scott Petersen
Study of the theoretical and practical fundamentals of computer-generated music, with a focus on low-level sound representation, acoustics and sound synthesis, scales and tuning systems, and programming languages for computer music generation. Theoretical concepts are supplemented with pragmatic issues expressed in a high-level programming language. Prerequisite: ability to read music.

**CPSC 535a, Building an Internet Router**  Robert Soule
Over the course of the term, students build a fully functioning Internet router. Students design the control plane in Python on a Linux host and design the data plane in the new P4 language on the bmv2 software switch. To provide context and background for the design of their router, students read a selection of papers to get both a historical perspective and exposure to current research in networking. Prerequisite: CPSC 533.

**CPSC 537a, Introduction to Database Systems**  Avi Silberschatz
CPSC 539a or b, Software Engineering  Timos Antonopoulos
Introduction to building a large software system in a team. Learning how to collect requirements and write a specification. Project planning and system design. Increasing software reliability: debugging, automatic test generation. Introduction to type systems, static analysis, and model checking.

CPSC 546a, Data and Information Visualization  Holly Rushmeier
Visualization is a powerful tool for understanding data and concepts. This course provides an introduction to the concepts needed to build new visualization systems, rather than to use existing visualization software. Major topics are abstracting visualization tasks, using visual channels, spatial arrangements of data, navigation in visualization systems, using multiple views, and filtering and aggregating data. Case studies to be considered include a wide range of visualization types and applications in humanities, engineering, science, and social science. Prerequisite: CPSC 223.

CPSC 547a, Introduction to Quantum Computing  Yongshan Ding
This course introduces the fundamental concepts in the theory and practice of quantum computing. Topics covered include information processing, quantum programming, quantum compilation, quantum algorithms, and error correction. The objective of the course is to engage students in applying fresh thinking to what computers can do. We establish an understanding of how quantum computers store and process data, and we discover how they differ from conventional digital computers. We anticipate this course will be of interest to students working in computer science, electrical engineering, physics, or mathematics. Students must be comfortable with programming, discrete probability, and linear algebra. Prior experience in quantum computing is useful but not required.

CPSC 551b, The User Interface  David Gelernter
The user interface (UI) in the context of modern design, where tech has been a strong and consistent influence from the Bauhaus and U.S. industrial design of the 1920s and 1930s through the IBM-Eames design project of the 1950s to 1970s. The UI in the context of the windows-menus-mouse desktop, as developed by Alan Kay and Xerox in the 1970s and refined by Apple in the early 1980s. Students develop a detailed design and simple implementation for a UI.

CPSC 552b / AMTH 552b / CB&B 663b, Deep Learning Theory and Applications  Smita Krishnaswamy
Deep neural networks have gained immense popularity within the past decade due to their success in many important machine-learning tasks such as image recognition, speech recognition, and natural language processing. This course provides a principled and hands-on approach to deep learning with neural networks. Students master the principles and practices underlying neural networks, including modern methods of deep learning, and apply deep learning methods to real-world problems including image recognition, natural language processing, and biomedical applications. Course work includes homework, a final exam, and a final project—either group or individual, depending on enrollment—with both a written and oral (i.e., presentation) component. The course assumes basic prior knowledge in linear algebra and probability. Prerequisites: CPSC 202 and knowledge of Python programming.
CPSC 554b, Software Analysis and Verification  Ruzica Piskac
Introduction to concepts, tools, and techniques used in the formal verification of software. State-of-the-art tools used for program verification; detailed insights into algorithms and paradigms on which those tools are based, including model checking, abstract interpretation, decision procedures, and SMT solvers.

CPSC 555a, Economics and Computation  Yang Cai
A mathematically rigorous investigation of the interplay of economic theory and computer science, with an emphasis on the relationship of incentive-compatibility and algorithmic efficiency. Particular attention to the formulation and solution of mechanism-design problems that are relevant to data networking and Internet-based commerce.

CPSC 558b, Automated Decision Systems  Stephen Slade
People make dozens of decisions every day in their personal and professional lives. What would it mean for you to trust a computer to make those decisions for you? It is likely that many of those decisions are already informed, mediated, or even made by computer systems. Explicit examples include dating sites like match.com or recommendation systems such as Amazon or Netflix. Most Internet ads on sites like Google or Facebook are run by real-time-bidding (RTB) systems that conduct split-second auctions in the hopes of getting your attention. Driverless cars offer the promise of safer highways. Corporations and other enterprises invest in decision support systems to improve the quality of their products and services. This course considers the spectrum of automated decision models and tools, examining their costs and effectiveness. Examples come from a variety of fields including finance, risk management, credit-card fraud, robotics, medicine, and politics.

CPSC 559a, Building Interactive Machines  Marynel Vazquez
This advanced course brings together methods from machine learning, computer vision, robotics, and human–computer interaction to enable interactive machines to perceive and act in a variety of environments. Part of the course examines approaches for perception with different sensing devices and algorithms; the other part focuses on methods for decision-making and applied machine learning for control. The course is a combination of lectures, state-of-the-art reading, presentations and discussions, programming assignments, and a final team project. Prerequisites: CPSC 570 and understanding of probability, differential calculus, linear algebra, and planning (in Artificial Intelligence). Programming assignments require proficiency in Python and high-level familiarity with C++. Students who do not fit this profile may be allowed to enroll with the permission of the instructor.

CPSC 564a, Algorithms and their Societal Implications  Nisheeth Vishnoi
Today’s society comprises humans living in an interconnected world that is intertwined with a variety of sensing, communicating, and computing devices. Human-generated data is being recorded at unprecedented rates and scales, and powerful AI and ML algorithms, which are capable of learning from such data, are increasingly controlling various aspects of modern society: from social interactions. These data-driven decision-making algorithms have a tremendous potential to change our lives for the better, but, via the ability to mimic and nudge human behavior, they also have the potential to be discriminatory, reinforce societal prejudices, violate privacy, polarize opinions, and influence democratic processes. Thus, designing effective tools to govern modern society which reinforce its cherished values such as equity, justice, democracy, health,
privacy, etc. has become paramount and requires a foundational understanding of how humans, data, and algorithms interact. This course is for students who would like to understand and address some of the key challenges and emerging topics at the aforementioned interplay between computation and society. On the one hand, we study human decision-making processes and view them through the lens of computation, and on the other hand we study and address the limitations of artificial decision-making algorithms when deployed in various societal contexts. The focus is on developing solutions through a combination of foundational work such as coming up with the right definitions, modeling, algorithms, and empirical evaluation. The current focus is on bias and privacy, with additional topics including robustness, polarization, and democratic representation. The grade will be based on class participation and a project. The project grade will be determined by a midterm and endterm report/presentation. The course has four primary modules: (1) Data: human-generated data; data collection and aggregation; (2) Decision-Making Algorithms: human decision-making algorithms; traditional algorithmic decision-making models and methods; machine learning-based decision-making models and methods; (3) Bias: socio-technical contexts and underlying computational problems; definitions of fairness; interventions for ensuring fairness; human biases through the lens of computation; privacy; need for definitions of privacy; differential privacy; beyond differential privacy; (4) Other topics: robustness; polarization; elections and social choice. Solid mathematical and programming background is necessary to enroll in this course. CPSC 365 and S&DS 251 are recommended.

**CPSC 565a, Theory of Distributed Systems** James Aspnes
Models of asynchronous distributed computing systems. Fundamental concepts of concurrency and synchronization, communication, reliability, topological and geometric constraints, time and space complexity, and distributed algorithms.

**CPSC 567b, Cryptography and Computer Security** Charalampos Papamanthou
A survey of such private and public key cryptographic techniques as DES, RSA, and zero-knowledge proofs, and their application to problems of maintaining privacy and security in computer networks. Focus on technology, with consideration of such societal issues as balancing individual privacy concerns against the needs of law enforcement, vulnerability of societal institutions to electronic attack, export regulations and international competitiveness, and development of secure information systems.

**CPSC 568a, Computational Complexity** Staff
Introduction to the theory of computational complexity. Basic complexity classes, including polynomial time, nondeterministic polynomial time, probabilistic polynomial time, polynomial space, logarithmic space, and nondeterministic logarithmic space. The roles of reductions, completeness, randomness, and interaction in the formal study of computation.

**CPSC 569b, Randomized Algorithms** James Aspnes
Beginning with an introduction to tools from probability theory including some inequalities like Chernoff bounds, the course covers randomized algorithms from several areas: graph algorithms, algorithms in algebra, approximate counting, probabilistically checkable proofs, and matrix algorithms.
CPSC 570b, Artificial Intelligence  Staff
Introduction to artificial intelligence research, focusing on reasoning and perception. Topics include knowledge representation, predicate calculus, temporal reasoning, vision, robotics, planning, and learning.

CPSC 572a, Intelligent Robotics  Brian Scassellati
Introduction to the construction of intelligent, autonomous systems. Sensory-motor coordination and task-based perception. Implementation techniques for behavior selection and arbitration, including behavior-based design, evolutionary design, dynamical systems, and hybrid deliberative-reactive systems. Situated learning and adaptive behavior.

CPSC 573b, Intelligent Robotics Laboratory  Brian Scassellati
Students work in small teams to construct novel research projects using one of a variety of robot architectures. Project topics may include human-robot interaction, adaptive intelligent behavior, active perception, humanoid robotics, and socially assistive robotics.

CPSC 574a, Computational Intelligence for Games  James Glenn
A seminar on current topics in computational intelligence for games, including developing agents for playing games, procedural content generation, and player modeling. Students read, present, and discuss recent papers and competitions, and complete a term-long project that applies some of the techniques discussed during the term to a game of their choice.

CPSC 575a / ENAS 575a / INP 575a, Computational Vision and Biological Perception  Steven Zucker
An overview of computational vision with a biological emphasis. Suitable as an introduction to biological perception for computer science and engineering students, as well as an introduction to computational vision for mathematics, psychology, and physiology students.

CPSC 577b, Natural Language Processing  Dragomir Radev
Linguistic, mathematical, and computational fundamentals of natural language processing (NLP). Topics include part of speech tagging, Hidden Markov models, syntax and parsing, lexical semantics, compositional semantics, machine translation, text classification, discourse, and dialogue processing. Additional topics such as sentiment analysis, text generation, and deep learning for NLP.

CPSC 578b, Computer Graphics  Julie Dorsey
Introduction to the basic concepts of two- and three-dimensional computer graphics. Topics include affine and projective transformations, clipping and windowing, visual perception, scene modeling and animation, algorithms for visible surface determination, reflection models, illumination algorithms, and color theory.

CPSC 579a, Advanced Topics in Computer Graphics  Julie Dorsey
An in-depth study of advanced algorithms and systems for rendering, modeling, and animation in computer graphics. Topics vary and may include reflectance modeling, global illumination, subdivision surfaces, NURBS, physically based fluids systems, and character animation.
CPSC 582b, Current Topics in Applied Machine Learning  David van Dijk
We cover recent advances in machine learning that focus on real-world data. We discuss a wide range of methods and their applications to diverse domains, such as finance, health care, genomics, protein folding, drug discovery, neuroscience, and natural language processing. The seminar is based on a series of lectures by the instructor and guest lecturers, as well as student presentations. The latter are expected to be on recent publications from leading journals and conferences in the field and are followed by discussions. A final project involves the application of a machine-learning method to real-world data. Graduate students are required to work on projects, which are optional for undergraduates. Prerequisites: mathematical tools for computer science (CPSC 202 or equivalent course), linear algebra (MATH 222/MATH 225 or equivalent course), calculus (MATH 120 or equivalent course), or permission of the instructor; and basic coding knowledge (e.g., Python).

CPSC 583a, Deep Learning on Graph-Structured Data  Rex Ying
Graph structure emerges in many important domain applications, including but not limited to computer vision, natural sciences, social networks, languages, and knowledge graphs. This course offers an introduction to deep learning algorithms applied to such graph-structured data. The first part of the course is an introduction to representation learning for graphs and covers common techniques in the field, including distributed node embeddings, graph neural networks, deep graph generative models, and non-Euclidean embeddings. The first part also touches upon topics of real-world significance, including auto-ML and explainability for graph learning. The second part of the course covers important applications of graph machine learning. We learn ways to model data as graphs and apply graph learning techniques to problems in domains including online recommender systems, knowledge graphs, biological networks, physical simulations and graph mining. The course covers many deep techniques (graph neural networks, graph deep generative models) catered to graph structures. We cover basic deep learning tutorials in this course. Knowledge of graphs as a data structure, and understanding of basic graph algorithms are essential for applying machine learning to graph-structured data. Familiarity with Python and important libraries such as Numpy and Pandas are helpful. A foundation of deep neural networks is highly recommended. Experience in machine Learning and Graph Theory are welcomed as well.

CPSC 584b, Introduction to Human-Computer Interaction  Marynel Vazquez
This course introduces students to the interdisciplinary field of human-computer interaction (HCI), with particular focus on human-robot interaction (HRI). The first part of the course covers principles and techniques in the design, development, and evaluation of interactive systems. It provides students with an introduction to UX design and user-centered research. The second part focuses on the emergent field of HRI and several other nontraditional interfaces, e.g., AR/VR, tangibles, crowdsourcing. The course is organized as a series of lectures, presentations, a midterm exam, and a term-long group project on designing a new interactive system. Prerequisites: CPSC 201 and CPSC 202 or equivalents. Students who do not fit this profile may be allowed to enroll with permission of the instructor.
CPSC 611a, Topics in Computer Science and Global Affairs  Joan Feigenbaum and Ted Wittenstein
This course focuses on “socio-technical” problems in computing and international relations. These are problems that cannot be solved through technological progress alone but rather require legal, political, or cultural progress as well. Examples include but are not limited to cyber espionage, disinformation, ransomware attacks, and intellectual-property theft. This course is offered jointly by the SEAS Computer Science Department and the Jackson School of Global Affairs. It is addressed to graduate students who are interested in socio-technical issues but whose undergraduate course work may not have addressed them; it is designed to bring these students rapidly to the point at which they can do research on socio-technical problems. Prerequisites: Basics of cryptography and computer security (as covered in Yale’s CPSC 467), networks (as covered in Yale’s CPSC 433), and databases (as covered in Yale’s CPSC 437) helpful but not required.

CPSC 640b / AMTH 640b, Topics in Numerical Computation  Vladimir Rokhlin
This course discusses several areas of numerical computing that often cause difficulties to non-numericists, from the ever-present issue of condition numbers and ill-posedness to the algorithms of numerical linear algebra to the reliability of numerical software. The course also provides a brief introduction to “fast” algorithms and their interactions with modern hardware environments. The course is addressed to Computer Science graduate students who do not necessarily specialize in numerical computation; it assumes the understanding of calculus and linear algebra and familiarity with (or willingness to learn) either C or FORTRAN. Its purpose is to prepare students for using elementary numerical techniques when and if the need arises.

CPSC 644a / MATH 522a, Geometric and Topological Methods in Machine Learning  Smita Krishnaswamy and Ian Adelstein
This course provides an introduction to geometric and topological methods in data science. Our starting point is the manifold hypothesis: that high dimensional data live on or near a much lower dimensional smooth manifold. We introduce tools to study the geometric and topological properties of this manifold in order to reveal relevant features and organization of the data. Topics include: metric space structures, curvature, geodesics, diffusion maps, eigenmaps, geometric model spaces, gradient descent, data embeddings and projections, and topological data analysis (TDA) in the form of persistence homology and their associated “barcodes.” We see applications of these methods in a variety of data types. Prerequisites: MATH 225 or 226; MATH 255 or 256; MATH 302 and CPSC 112. Students who completed MATH 231 or 250 may substitute another analysis course level 300 or above in place of MATH 302. Familiarity with algorithms/programming is beneficial.

CPSC 668a, Frontiers of Blockchain Research  Benjamin Fisch
This course engages students with research problems pertinent to blockchain systems, such as Bitcoin and Ethereum, spanning a wide variety of topics including privacy, scalability, verifiability, decentralization, interoperability, and economics. The course begins with a detailed survey of specific tools from cryptography and distributed systems, such as authenticated data structures, proof systems, and consensus protocols while focusing on how these tools are combined to achieve the desiderata of blockchains. We then center the discussion around recent research papers and open problems. The course examines both theoretical and applied aspects.
of blockchain systems, from formal security models and analysis to lessons and observations drawn from how these systems have behaved in practice. There are no required textbooks. Readings will be posted. The course grading will be based on participation in discussions and a research project/report. Open to Ph.D. and M.S. students in computer science; advanced undergraduates may enroll with permission of the instructor. The course assumes background in various fundamental areas of CS, including discrete math, probability, algorithms, data structures, networks, and distributed systems. Background in cryptography and computer security is highly recommended (e.g., CPSC 567 or equivalent experience). Students who are confident in their ability to read and digest two papers a week from venues such as Usenix Security and IEEE S&P should be able to keep pace with the course.

**CPSC 672a, Interactive Robot Learning**  Staff
This class explores methods for grounding machine learning problems in human-robot interaction (HRI). We cover topics including learning from demonstration, active learning, representations for modeling high-level and low-level task information, and human factors for designing learning interactions. Students are asked to read and present research papers on these topics from top publication venues in machine learning and HRI. Students also complete a semester-long research project that implements and evaluates state-of-the-art methods for interactive robot learning. This course is aimed at Ph.D. and M.S. students and, with the permission of the instructor, undergraduates with a background in artificial intelligence. In particular, students must have successfully passed an AI course such as CPSC 470/570.

**CPSC 676a, Advanced Computer Vision**  Staff
This course focuses on advanced topics in computer vision. In particular, we cover topics such as camera model and calibration, image descriptors, sparse correspondences, perspective projection, projective transformation, epipolar geometry, and 3D reconstruction with stereo and structure-from-motion. It also covers learning-based methods for depth estimation from sensor fusion, stereo, and their robustness under various adverse conditions. Students read, present, and lead discussions on research papers published at top-tier conferences and journals in the field of computer vision. Students undertake a semester-long research project related to the topics discussed in the course. Successful projects can be submitted to peer-reviewed conferences and journals with the aim of publication. This course is aimed at Ph.D. and M.S. students and, with the permission of the instructor, undergraduates with a background in computer vision and machine learning. Students taking this course must have successfully passed courses in object oriented programming (e.g., CPSC 112, CPSC 201, or equivalent courses), calculus (e.g., MATH 112, MATH 115, MATH 120, or equivalent courses), and linear algebra (e.g., MATH 225, or equivalent courses). A background in statistics, machine learning, and deep learning is useful but not required. Experience in programming with Python is preferable, as we use it for assignments and projects. Because the course is research-oriented, pre-requisites are strictly enforced.

**CPSC 679b, Physics Simulation for Movies and Games**  Theodore Kim
This course covers computational methods that commonly arise when simulating physics in movies and games. In particular, we learn state-of-the-art methods for simulating fluids (fire and water) and solids (muscles, clothing, and skin). The algorithms discussed span offline techniques suitable for movies and fabrication, as well as real-time techniques for games. We cover finite difference and finite
element representations as well as solver practicalities such as conjugate gradients, preconditioning, and Newton iteration. Prerequisites: linear algebra, Newtonian physics, and two terms of programming. Some familiarity with C, C++, or Java is assumed. Previous experience with computer graphics is preferred, but not required.

**CPSC 690a, Independent Project I**  Staff  
By arrangement with faculty.

**CPSC 691a, Independent Project II**  Staff  
By arrangement with faculty.

**CPSC 752b / CB&B 752b / MB&B 752b and MB&B 753b and MB&B 754b / MB&B 753b and MB&B 754b / MCDB 752b, Biomedical Data Science: Mining and Modeling**  Mark Gerstein  
Biomedical data science encompasses the analysis of gene sequences, macromolecular structures, and functional genomics data on a large scale. It represents a major practical application for modern techniques in data mining and simulation. Specific topics to be covered include sequence alignment, large-scale processing, next-generation sequencing data, comparative genomics, phylogenetics, biological database design, geometric analysis of protein structure, molecular-dynamics simulation, biological networks, normalization of microarray data, mining of functional genomics data sets, and machine-learning approaches to data integration. Prerequisites: biochemistry and calculus, or permission of the instructor.

**CPSC 990a, Ethical Conduct of Research for Master’s Students**  Inyoung Shin  
This course meets on four consecutive Fridays.

**CPSC 991a / MATH 991a, Ethical Conduct of Research**  Holly Rushmeier  
0 Course cr

**CPSC 992a, Academic Writing**  Staff  
This course is an intensive analysis of the principles of excellent writing for Ph.D. students and scientists preparing a range of texts including research papers, conference posters, technical reports, research statements, grant proposals, correspondence, science and industry blogs, and other relevant documents. We look at the components of rhetorical positioning in the development of a clear, interesting, and rigorous science research paper. Some of the sub-genres we analyze and practice include the introduction, literature review, methodology, data commentary, results/discussion, conclusion, and abstract. In addition to the research paper, we practice other types of texts including research statements, requests for funding, bio-data statements, and blogs. We also discuss how writers can develop content and fluency as well as strategies for redrafting and editing. Students receive detailed feedback on their writing with a focus on clarity, precision, tone, and readability. 0 Course cr